CLAIMS

What is claimed is:

1. A method of dynamic installation and activation of software packages in a node in a distributed network of nodes, the method comprising the computer-implemented steps of:

providing a master node;

providing software package storage means on said master node for storing software packages and software modules that the nodes in the network will be using as well as older versions that are kept for regressing a node back to a previous module or software package version;

wherein a software package contains at least one module and its associated dependency information;

receiving a software update for a node on said master node;

wherein the software update contains a software package, or a set of software packages;

storing the software update on said software package storage means;

wherein said master node notifies said node that a software update is being requested; and

wherein said master node passes said node identities of software package(s) to be updated and module dependencies.

2. A method as recited in Claim 1, wherein each module has a binary signature;.

3. A method as recited in Claim 1, wherein each node has a list of desired characteristics stored on said master node which is compared by said master node to each module in the software update to determine which subset of modules should be sent to a node.

4. A method as recited in Claim 1, wherein said node determines running processes on said node that will be affected by the software update using the module dependencies.

5. A method as recited in Claim 4, wherein said node notifies affected processes that the software update is being requested; wherein each notified process evaluates the effect that the software update will have on its operation; wherein if any of the notified processes determine that the software update will degrade or have a negative impact on said node's normal operation, the process returns a veto to said node; and wherein if a process finds that the software update will have no negative effects, the process returns an acceptance of the software update to said node.

6. A method as recited in Claim 5, wherein said node waits for all of the notified processes to return the results of their evaluations and once all of the processes have reported to said node, said node notifies said master node if any of the processes have vetoed the software update.

7. A method as recited in Claim 6, wherein if said master node receives an acceptance from said node then said master node sends appropriate software package(s) for the software update from said software package storage means to said node.

8. A method as recited in Claim 7, wherein said node immediately runs software package modules, by loading the modules from the software package(s) and signals processes that are being replaced by the modules and the affected processes that the changeover is going to occur; wherein when all of the signaled processes indicate that they are ready and waiting for the changeover, said node starts new modules and signals the affected processes that the changeover has occurred; wherein each module starts without affecting normal operation of said node; and wherein each affected process restarts, if required, without affecting normal operation of said node.

9. A method as recited in Claim 8, wherein said node continues with normal operations and notifies said master node that it has completed the software update; and wherein said master node checks the dependencies of the software package(s) for the software update to ensure that any inter-nodal and intra-node dependencies are complete.

10. A method as recited in Claim 9, wherein if there are any discrepancies in the inter-nodal and intra-node dependencies, then said master node notifies a user.

11. A method as recited in Claim 8, wherein when said node does not store the software package(s) in its local persistent storage, then said node can later regress back to previous modules stored in the local persistent storage if it restarts or said master node tells it to regress.

12. A method as recited in Claim 7, wherein said node extracts version information and dependency information from the software package(s) and stores the information in its local persistent storage.

13. A method as recited in Claim 12, wherein said node compares binary signatures of the modules in the software package(s) with corresponding modules stored in the local persistent storage to discover which modules have been updated; wherein any binary signatures that match indicate that the module has not changed; and wherein any modules that have different binary signatures replace the corresponding modules stored in the local persistent storage.

14. A method as recited in Claim 13, wherein said node runs software package modules by loading the modules from the software package(s) and signals processes that are being replaced by the modules and the affected processes that the changeover is going to occur; wherein when all of the signaled processes indicate that they are ready and waiting for the changeover, the node starts new modules and signals the affected processes that the changeover has occurred; wherein each module starts without affecting normal operation of said node; and wherein each affected process restarts, if required, without affecting normal operation of said node.

15. A method as recited in Claim 14, wherein said node continues with normal operations and notifies said master node that it has completed the software update; and wherein said master node checks the dependencies of the software package(s) for the software update to ensure that any inter-nodal and intra-node dependencies are complete.

16. A method as recited in Claim 15, wherein if there are any discrepancies in the inter-nodal and intra-node dependencies, then said master node notifies a user.

17. A method as recited in Claim 6, wherein if more than one node was being updated, the software update will not occur if any node vetoes the software update.

18. A method as recited in Claim 6, wherein if said master node receives a veto from said node, then said master node does not update said node and notifies a user that the software update will adversely affect said node; and wherein the user must then make a decision whether to update some or all of the nodes, or to abort the update.

19. A method as recited in Claim 18, wherein if the user decides to continue updating said node, then said master node forces said node to accept the software update.

20. A method as recited in Claim 1, wherein a user initiates a software update by installing an image containing the software update onto said master node.

21. A method as recited in Claim 20, wherein the user indicates what nodes and which software package(s) are to be updated.

22. A method as recited in Claim 1, wherein the software update contains a list of nodes to be updated.

23. A method as recited in Claim 1, wherein the software update contains a list of software packages destined for each node.

24. A method as recited in Claim 1, wherein the master node has the ability to categorize nodes into classes where all of the nodes in a particular class of nodes have the same software configuration and may have differing processor types.

25. A method as recited in Claim 1, wherein a software package contains version information, dependency information, and other metadata information pertaining to software in the package.

26. A method as recited in Claim 25, wherein the metadata includes a list of application program interface (API) providers and consumers.

27. A computer-readable medium carrying one or more sequences of instructions for dynamic installation and activation of software packages in a node in a distributed network of nodes, which instructions, when executed by one or more processors, cause the one or more processors to carry out the steps of:

providing a master node;

providing software package storage means on said master node for storing software packages and software modules that the nodes in the network will be using as well as older versions that are kept for regressing a node back to a previous module or software package version;

wherein a software package contains at least one module;

receiving a software update for a node on said master node;

wherein the software update contains a software package, or a set of software packages;

storing the software update on said software package storage means;

wherein said master node notifies said node that a software update is being requested; and

wherein said master node passes said node identities of software package(s) to be updated and module dependencies.

28. A computer-readable medium as recited in Claim 27, wherein each module has a binary signature;.

29. A computer-readable medium as recited in Claim 27, wherein each node has a list of desired characteristics stored on said master node which is compared by said master node to each module in the software update to determine which subset of modules should be sent to a node.

30. A computer-readable medium as recited in Claim 27, wherein said node determines running processes on said node that will be affected by the software update using the module dependencies.

31. A computer-readable medium as recited in Claim 30, wherein said node notifies affected processes that the software update is being requested; wherein each notified process evaluates the effect that the software update will have on its operation; wherein if any of the notified processes determine that the software update will degrade or have a negative impact on said node's normal operation, the process returns a veto to said node; and wherein if a process finds that the software update will have no negative effects, the process returns an acceptance of the software update to said node.

32. A computer-readable medium as recited in Claim 31, wherein said node waits for all of the notified processes to return the results of their evaluations and once all of the processes have reported to said node, said node notifies said master node if any of the processes have vetoed the software update.

33. A computer-readable medium as recited in Claim 32, wherein if said master node receives an acceptance from said node then said master node sends appropriate software package(s) for the software update from said software package storage means to said node.

34. A computer-readable medium as recited in Claim 33, wherein said node immediately runs software package modules, by loading the modules from the software package(s) and signals processes that are being replaced by the modules and the affected processes that the changeover is going to occur; wherein when all of the signaled processes indicate that they are ready and waiting for the changeover, the node starts new modules and signals the affected processes that the changeover has occurred; wherein each module starts without affecting normal operation of said node; and wherein each affected process restarts, if required, without affecting normal operation of said node.

35. A computer-readable medium as recited in Claim 34, wherein said node continues with normal operations and notifies said master node that it has completed the software update; and wherein said master node checks the dependencies of the software package(s) for the software update to ensure that any inter-nodal and intra-node dependencies are complete.

36. A computer-readable medium as recited in Claim 35, wherein if there are any discrepancies in the inter-nodal and intra-node dependencies, then said master node notifies a user.

37. A computer-readable medium as recited in Claim 34, wherein when said node does not store the software package(s) in its local persistent storage, then said node can later regress back to previous modules stored in the local persistent storage if it restarts or said master node tells it to regress.

38. A computer-readable medium as recited in Claim 33, wherein said node extracts version information and dependency information from the software package(s) and stores the information in its local persistent storage.

39. A computer-readable medium as recited in Claim 38, wherein said node compares binary signatures of the modules in the software package(s) with corresponding modules stored in the local persistent storage to discover which modules have been updated; wherein any binary signatures that match indicate that the module has not changed; and wherein any modules that have different binary signatures replace the corresponding modules stored in the local persistent storage.

40. A computer-readable medium as recited in Claim 39, wherein said node runs software package modules by loading the modules from the software package(s) and signals processes that are being replaced by the modules and the affected processes that the changeover is going to occur; wherein when all of the signaled processes indicate that they are ready and waiting for the changeover, the node starts new modules and signals the affected processes that the changeover has occurred; wherein each module starts without affecting normal operation of said node; and wherein each affected process restarts, if required, without affecting normal operation of said node.

41. A computer-readable medium as recited in Claim 40, wherein said node continues with normal operations and notifies said master node that it has completed the software update; and wherein said master node checks the dependencies of the software package(s) for the software update to ensure that any inter-nodal and intra-node dependencies are complete.

42. A computer-readable medium as recited in Claim 41, wherein if there are any discrepancies in the inter-nodal and intra-node dependencies, then said master node notifies a user.

43. A computer-readable medium as recited in Claim 32, wherein if more than one node was being updated, the software update will not occur if any node vetoes the software update.

44. A computer-readable medium as recited in Claim 32, wherein if said master node receives a veto from said node, then said master node does not update said node and notifies a user that the software update will adversely affect said node; and wherein the user must then make a decision whether to update some or all of the nodes, or to abort the update.

45. A computer-readable medium as recited in Claim 44, wherein if the user decides to continue updating said node, then said master node forces said node to accept the software update.

46. A computer-readable medium as recited in Claim 27, wherein a user initiates a software update by installing an image containing the software update onto said master node.

47. A computer-readable medium as recited in Claim 46, wherein the user indicates what nodes and which software package(s) are to be updated.

48. A computer-readable medium as recited in Claim 27, wherein the software update contains a list of nodes to be updated.

49. A computer-readable medium as recited in Claim 27, wherein the software update contains a list of software packages destined for each node.

50. A computer-readable medium as recited in Claim 27, wherein the master node has the ability to categorize nodes into classes where all of the nodes in a particular class of nodes have the same software configuration and may have differing processor types.

51. A computer-readable medium as recited in Claim 27, wherein a software package contains version information, dependency information, and other metadata information pertaining to software in the package.

52. A computer-readable medium as recited in Claim 51, wherein the metadata includes a list of application program interface (API) providers and consumers

53. An apparatus of dynamic installation and activation of software packages in a node in a distributed network of nodes, comprising:

a master node;

software package storage means on said master node for storing software packages and software modules that the nodes in the network will be using as well as older versions that are kept for regressing a node back to a previous module or software package version;

wherein a software package contains at least one module;

means for receiving a software update for a node on said master node;

wherein the software update contains a software package, or a set of software packages;

means for storing the software update on said software package storage means;

wherein said master node notifies said node that a software update is being requested; and

wherein said master node passes said node identities of software package(s) to be updated and module dependencies.

54. An apparatus as recited in Claim 53, wherein each module has a binary signature;.

55. An apparatus as recited in Claim 53, wherein each node has a list of desired characteristics stored on said master node which is compared by said master node to each module in the software update to determine which subset of modules should be sent to a node.

56. An apparatus as recited in Claim 53, wherein said node determines running processes on said node that will be affected by the software update using the module dependencies.

57. An apparatus as recited in Claim 56, wherein said node notifies affected processes that the software update is being requested; wherein each notified process evaluates the effect that the software update will have on its operation; wherein if any of the processes determine that the software update will degrade or have a negative impact on said node's normal operation, the process returns a veto to said node; and wherein if a process finds that the software update will have no negative effects, the process returns an acceptance of the software update to said node.

58. An apparatus as recited in Claim 57, wherein said node waits for all of the notified processes to return the results of their evaluations and once all of the processes have reported to said node, said node notifies said master node if any of the processes have vetoed the software update.

59. An apparatus as recited in Claim 58, wherein if said master node receives an acceptance from said node then said master node sends appropriate software package(s) for the software update from said software package storage means to said node.

60. An apparatus as recited in Claim 59, wherein said node immediately runs software package modules, by loading the modules from the software package(s) and signals processes that are being replaced by the modules and the affected processes that the changeover is going to occur; wherein when all of the signaled processes indicate that they are ready and waiting for the changeover, the node starts new modules and signals the affected processes that the changeover has occurred; wherein each module starts without affecting normal operation of said node; and wherein each affected process restarts, if required, without affecting normal operation of said node.

61. An apparatus as recited in Claim 60, wherein said node continues with normal operations and notifies said master node that it has completed the software update; and wherein said master node checks the dependencies of the software package(s) for the software update to ensure that any inter-nodal and intra-node dependencies are complete.

62. An apparatus as recited in Claim 61, wherein if there are any discrepancies in the inter-nodal and intra-node dependencies, then said master node notifies a user.

63. An apparatus as recited in Claim 60, wherein when said node does not store the software package(s) in its local persistent storage, then said node can later regress back to previous modules stored in the local persistent storage if it restarts or said master node tells it to regress.

64. An apparatus as recited in Claim 59, wherein said node extracts version information and dependency information from the software package(s) and stores the information in its local persistent storage.

65. An apparatus as recited in Claim 64, wherein said node compares binary signatures of the modules in the software package(s) with corresponding modules stored in the local persistent storage to discover which modules have been updated; wherein any binary signatures that match indicate that the module has not changed; and wherein any modules that have different binary signatures replace the corresponding modules stored in the local persistent storage.

66. An apparatus as recited in Claim 65, wherein said node runs software package modules by loading the modules from the software package(s) and signals processes that are being replaced by the modules and the affected processes that the changeover is going to occur; wherein when all of the signaled processes indicate that they are ready and waiting for the changeover, the node starts new modules and signals the affected processes that the changeover has occurred; wherein each module starts without affecting normal operation of said node; and wherein each affected process restarts, if required, without affecting normal operation of said node.

67. An apparatus as recited in Claim 66, wherein said node continues with normal operations and notifies said master node that it has completed the software update; and wherein said master node checks the dependencies of the software package(s) for the software update to ensure that any inter-nodal and intra-node dependencies are complete.

68. An apparatus as recited in Claim 67, wherein if there are any discrepancies in the inter-nodal and intra-node dependencies, then said master node notifies a user.

69. An apparatus as recited in Claim 58, wherein if more than one node was being updated, the software update will not occur if any node vetoes the software update.

70. An apparatus as recited in Claim 58, wherein if said master node receives a veto from said node, then said master node does not update said node and notifies a user that the software update will adversely affect said node; and wherein the user must then make a decision whether to update some or all of the nodes, or to abort the update.

71. An apparatus as recited in Claim 70, wherein if the user decides to continue updating said node, then said master node forces said node to accept the software update.

72. An apparatus as recited in Claim 53, wherein a user initiates a software update by installing an image containing the software update onto said master node.

73. An apparatus as recited in Claim 72, wherein the user indicates what nodes and which software package(s) are to be updated.

74. An apparatus as recited in Claim 53, wherein the software update contains a list of nodes to be updated.

75. An apparatus as recited in Claim 53, wherein the software update contains a list of software packages destined for each node.

76. An apparatus as recited in Claim 53, wherein the master node has the ability to categorize nodes into classes where all of the nodes in a particular class of nodes have the same software configuration and may have differing processor types.

77. An apparatus as recited in Claim 53, wherein a software package contains version information, dependency information, and other metadata information pertaining to software in the package.

78. An apparatus as recited in Claim 77, wherein the metadata includes a list of application program interface (API) providers and consumers.